ARM Processor
Cortex™-A15 MPCore-NEON (MP009)

Product Revision r4

Software Developers Errata Notice

Non-Confidential - Released

ARM-EPM-045620 v4.0 Released

Software Developers Errata Notice

Non-Confidential Proprietary Notice

Web Address

http://www.arm.com

Feedback on content

If you have any comments on content, then send an e-mail to errata@arm.com . Give:

- the document title
- the document number, ARM-EPM-045620
- the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Release Information

Errata are listed in this section if they are new to the document, or marked as "updated" if there has been any change to the erratum text in Chapter 2. Fixed errata are not shown as updated unless the erratum text has changed. The summary table in section 2.2 identifies errata that have been fixed in each product revision.

01 Nov 2013: Changes in Document v4

Page	Status	ID	Cat	Rare	Summary of Erratum
10	New	813469	CatB		An unaligned store instruction crossing a 4k page boundary at the same time as the lower page is invalidated might stall

25Sept 2013: Changes in Document v3

Page	Status	ID	Cat	Rare	Summary of Erratum
13	New	812169	CatB	Rare	DVM message blocked by copyback on AMBA Chi interconnect

21 Aug 2013: Changes in Document v2

Page	Status	ID	Cat	Rare	Summary of Erratum
16	New	810072	CatC		When a single-bit ECC error occurs in the L2, uncorrected data might be returned

10 Jul 2013: Changes in Document v1

Page	Status	ID	Cat	Rare	Summary of Erratum
8	New	784420	CatB		Speculative instruction fetches with MMU disabled might not comply with architectural requirements
11	New	763126	CatB	Rare	Three processor exclusive access livelock
9	New	785769	CatB		Undefined exception is not generated for LDC/STC instructions which access DCC registers in User mode
14	New	773023	CatC		Order may not be maintained between Strongly Ordered memory requests and Device memory requests on ACE/AXI
15	New	777769	CatC		ICache parity error may not be corrected for NC code

Contents

Chapter 1.
Introduction

This chapter introduces the errata notice for the ARM Cortex-A15 MPCore-NEON processor.

1.1. Scope of this document

This document describes errata categorized by level of severity. Each description includes:

- the current status of the defect
- where the implementation deviates from the specification and the conditions under which erroneous behavior occurs
- the implications of the erratum with respect to typical applications
- the application and limitations of a 'work-around' where possible

This document describes errata that may impact anyone who is developing software that will run on implementations of this ARM product.

1.2. Categorization of errata

Errata recorded in this document are split into the following levels of severity:

Table 1 Categorization of errata

Errata Type	Definition
Category A	A critical error. No workaround is available or workarounds are impactful. The error is likely to be common for many systems and applications.
Category A(rare)	A critical error. No workaround is available or workarounds are impactful. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category B	A significant error or a critical error with an acceptable workaround. The error is likely to be common for many systems and applications.
Category B(rare)	A significant error or a critical error with an acceptable workaround. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category C	A minor error.

Chapter 2.
Errata Descriptions

2.1. Product Revision Status

The r*n*p*n* identifier indicates the revision status of the product described in this book, where:

r*n* Identifies the major revision of the product.

p*n* Identifies the minor revision or modification status of the product.

2.2. Revisions Affected

Table 2 below lists the product revisions affected by each erratum. A cell marked with **X** indicates that the erratum affects the revision shown at the top of that column.

This document includes errata that affect revision r4 only.

Refer to the reference material supplied with your product to identify the revision of the IP.

Table 2 Revisions Affected

ID	Cat	Rare	Summary of Erratum	r4p0
784420	CatB		Speculative instruction fetches with MMU disabled might not comply with architectural requirements	X
785769	CatB		Undefined exception is not generated for LDC/STC instructions which access DCC registers in User mode	X
813469	CatB		An unaligned store instruction crossing a 4k page boundary at the same time as the lower page is invalidated might stall	X
763126	CatB	Rare	Three processor exclusive access livelock	X
812169	CatB	Rare	DVM message blocked by copyback on AMBA Chi interconnect	X
773023	CatC		Order may not be maintained between Strongly Ordered memory requests and Device memory requests on ACE/AXI	X
777769	CatC		ICache parity error may not be corrected for NC code	X
810072	CatC		When a single-bit ECC error occurs in the L2, uncorrected data might be returned	X

r4p0 implementation fix

Note the following errata may be fixed in some implementations of r4p0. This can be determined by reading the REVIDR register where a set bit indicates that the erratum is fixed in this part.

REVIDR[9:0]	Reserved
REVIDR[10]	810072 When a single-bit ECC error occurs in the L2, uncorrected data might be returned
REVIDR[11]	Reserved
REVIDR[12]	813469 An unaligned store instruction crossing a 4k page boundary at the same time as the lower page is invalidated might stall

Note that there is no change to the MIDR which remains at r4p0, but the REVIDR might be updated from 0x00 to indicate that one or more errata are corrected as shown above. Software will identify this release through the combination of MIDR and REVIDR.

2.3. Category A

2.4. Category A (Rare)

2.5. Category B

784420: Speculative instruction fetches with MMU disabled might not comply with architectural requirements

Category B

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r4p0

Description

When all applicable stages of translation are disabled, an ARMv7 processor must follow some architectural rules regarding speculative fetches and the addresses to which these can be initiated. These rules avoid potential reads to read-sensitive areas. For more information about these rules see the description of "Behavior of instruction fetches when all associated MMUs are disabled" in the ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition. Cortex-A15 normally operates with both the MMU and branch prediction enabled. If the processor operates in this condition for any significant amount of time, the BTB (branch target buffer) will contain branch predictions. If both stages of translation are then disabled, but branch prediction remains enabled, these stale BTB entries can cause A15 to violate the rules for speculative fetches.

Note: This erratum matches bug #5360 in the ARM internal Jira database.

Conditions

The erratum can occur only if the following sequence of conditions is met:

1) MMU enabled for at least one stage of address translation
2) Branch prediction enabled
3) Branches executed
4) MMU disabled for all applicable stages of address translation

Note: When executing in a Non-secure PL1 or PL0 mode, for condition 1 at least one stage of address translation must be enabled, and for condition 4 both stages of address translation must be disabled. When executing in any other mode, there is only one stage of address translation, that must be enabled for condition 1 and disabled for condition 4.

Implications

If the above conditions occur, it is possible that after the MMU is disabled, speculative instruction fetches will occur to read-sensitive locations.

Workaround

Branch prediction should be disabled when the MMU is disabled after having been enabled. This should be done by clearing the appropriate Z bit in the System Control register at the same time as or just before the final stage of translation is disabled. Branch prediction should remain disabled until the MMU is enabled, or until the BTB has been flushed. On A15, the BPI* branch predictor maintenance commands will not invalidate the BTB. The BTB can be flushed by setting bit 0 of the ACTLR register, doing any instruction cache invalidate instruction (e.g. ICIALLU), and then clearing bit 0 of the ACTLR register.

785769: Undefined exception is not generated for LDC/STC instructions which access DCC registers in User mode

Category B

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r4p0

Description

The Debug Communication Channel (DCC) registers are accessible using MCR/MRC instructions. LDC/STC instructions provide alternate access to DCC registers DBGDTRTXint/DBGDTRRXint.

In Non-debug state when DBGDSCR.UDCCdis is set to 1, then access to DCC register using MCR/MRC and LDC/STC instructions from User mode should generate an Undefined Instruction exception. Access using MCR/MRC instructions correctly generates an Undefined Instruction exception correctly. However access using LDC/STC instructions does not generate the Undefined Instruction exception and incorrectly accesses the register.

Note: This erratum matches bug #5372 in the ARM internal Jira database.

Conditions

1) The processor is in Non-debug state and User mode.
2) DBGDSCR.UDCCdis is set to 1.
3) Either the Hypervisor trap to debug registers in not set (HDCR.TDA==0) or the processor is in Secure state.
4) LDC to DBGDTRTXint or STC to DBGDTRRXint is executed.

Implications

If LDC or STC instructions are executed in User mode, then DCC traffic between debug host and debug target from FIQ/IRQ/Supervisor/Monitor/Abort/Hypervisor/Undefined/System modes can be corrupted due to this errata.

Workaround

Tools must use MCR/MRC instructions to access Debug Communication Channel (DCC) registers from User mode, instead of LDC/STC instructions, in Non-debug state.

Alternatively, software can avoid corruption of DCC traffic by Non-secure User mode code by setting the hypervisor trap for debug register accesses (HDCR.TDA), and handling the LDC/STC instruction appropriately in hypervisor code. There is no software workaround to prevent LDC/STC instructions executing in Secure User mode from corrupting DCC traffic handled in other processor modes.

813469: An unaligned store instruction crossing a 4k page boundary at the same time as the lower page is invalidated might stall

Category B

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r4p0.

Description

In a very unusual timing boundary condition, an unaligned store instruction crossing a 4k boundary at the same time as the lower page is being invalidated might stall until the next interrupt. If the store instruction is a VSTM of 88 bytes or larger, not 8-byte aligned, with 80 of the bytes in the lower page, the stall will not be broken by an interrupt and the core will stall until the next reset.

Note: This erratum matches bug #5469 in the ARM internal Jira database.

Conditions

1) A store is executed that crosses a 4k boundary.
2) The store has the following alignment:
   - 2-byte store, not 2-byte aligned
   - 4-byte store, not 4-byte aligned
   - 8-byte or larger store, not 8-byte aligned
3) The lower page hits a valid translation in the L1DTLB.
4) The upper page misses in the L1DTLB and the table walk returns a translation fault.
5) While the TLB miss for the upper page is being serviced, the lower page translation is invalidated by a TLBI instruction from an ARM core.
6) The lower page misses in the L1DTLB and the table walk returns a translation fault.

- All of the above conditions are required to hit the basic erratum, which will generally be broken by the next interrupt. To hit the stall that will not be broken by an interrupt, further conditions are required:

7) The store is a VSTM of 88 or more bytes, 4-byte aligned but not 8-byte aligned.
8) The first 80 bytes of the VSTM are in the lower page, with some of the bytes in the upper page.

Implications

If conditions 1-6 above occur with the correct timing, the CPU will stall until the next interrupt. If conditions 1-8 occur, that CPU will stall indefinitely, broken only by resetting that CPU.

The erratum will only occur in rare boundary conditions when an OS is invalidating a page that is actively being used by a process on another CPU and all the conditions are met. Because hitting the erratum is expected to be quite rare, in a system where the A15 CPUs get periodic interrupts the impact in most systems is likely to be minimal.

Compilers are not expected to generate an unaligned VSTM large enough to cause the indefinite stall. Optimized memcpy using Neon registers uses VSTR, not VSTM. Function prologues may use VSTM, but are not expected to store more than 64 bytes with normal calling conventions, even if the stack were not 8 byte aligned. Large VSTMs might occur in cases such as exception unwinding and state saving, but in those cases the VSTM will be 8-byte aligned and not affected. The VSTM indefinite stall case is therefore not expected to occur in a typical system.

Workaround

The suggested workaround is to avoid the use of large unaligned VSTM and to supply periodic interrupts to each CPU to clear the stall should it ever occur.

If large unaligned VSTM cannot be avoided, interrupts are not available, or a rare delay until the next interrupt is not acceptable, an alternative workaround is to do local TLB maintenance on each CPU rather than using the distributed TLB maintenance mechanism:

- Do TLB maintenance locally on each CPU using the local (not “inner shared”) TLB maintenance operations.

- Between TLB maintenance operations and the subsequent required DSB instruction, do not execute unaligned page crossing stores to the pages being invalidated.

2.6. Category B (Rare)

763126: Three processor exclusive access livelock

Category B Rare
Products Affected: Cortex-A15 MP Core -NEON.
Present in: r4p0

Description

In a system with three or more coherent masters that all use the ldrex/strex synchronization primitives to access a semaphore in coherent cacheable memory, there is a possibility of a livelock condition where two masters continuously attempt and fail to get the lock while the third master continuously reads the lock.

This erratum is heavily dependent on a unique set of initial conditions, and upon specific interconnect timing once the livelock has started. It is expected to be rare in a real system that the timing conditions will be hit.

An example: two cores C1 and C2 are contending for a lock using ldrex/strex, and core C3 is looping reading the same semaphore location. Once the livelock condition has started, from the perspective of C1, the sequence will look like this:

1) Execute ldrex, hits the cache in unique state.
2) External snoop takes line to shared state (triggered by C3 read).
3) Execute instructions to process the ldrex result and prepare the strex data.
4) Execute strex, hits cache shared, issues readUnique to bring in line unique.
5) External snoop invalidates line, clearing monitor (triggered by C2 strex that will eventually fail the monitor).
6) Line returns in unique state, but strex fails due to cleared monitor.
7) Loop back to step 1.

C1 and C2 constantly issue ReadUniques due to failing store exclusives that invalidate the line in the other core, each core causing the others strex to fail without making forward progress. No forward progress is made until/unless one of the cores stops (possibly due to an interrupt) or interconnect timing happens to allow enough time for one of them to complete.

NOTE: this erratum is describing additional limitations on exclusives loads and stores in a multi-core system including A15. There is no plan to fix this erratum on future A15 cores, as reasonable code following the ARM architecture guidelines should not be affected.

NOTE: This erratum matches bug #4637 in the ARM internal Jira database.

Conditions

1) One master continuously reading the location of the semaphore.
2) Two masters doing a ldrex/strex loop to the semaphore.
3) Semaphore in write-back shared memory.
4) Three master system (3+ core A15, or 3+ total processors in the system over ACE).

Implications

Neither C1 nor C2 will ever succeed in gaining the lock. Software could stop making progress. An interrupt to one of the cores C1/C2/C3 would likely break the livelock.

Workaround

If there are no more than two coherent masters in the system, no workaround is needed, the issue will not be seen.

The latest version of the ACE specification adds additional command types and system logic to allow processors to avoid this issue. This specification update was not available in time for A15 to take advantage of it and A15 does not implement this ACE feature. As an alternative, A15 installed hardware in each processor to detect that the load/store exclusive livelock scenario may be occurring and delay snoops for a period of time to allow the load exclusive/store

exclusive loop to complete and make forward progress. With this fix, no existing code that uses ldrex/strex should need to be rewritten if it follows the ARM Architecture Reference Manual guidelines in the “A3.4 Synchronization and Semaphores” section and is not unreasonably long.

To enable this hardware on Cortex-A15 you must set the "Snoop-delayed exclusive handling" bit in the Auxiliary Control Register, ACTLR[31] to 1. The reset value of ACTLR[31] is 0 for all product revisions r0pX, r1pX, r2pX, r3p0, r3p1 and r3p2. This reset value is 1 for product revision r3p3 and beyond.

Note: all references to “ldrex” encompass all Load-Exclusive instructions and “strex” encompass all Store-Exclusive instructions.

812169: DVM message blocked by copyback on AMBA Chi interconnect

Category B Rare

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r4p0

Description

This erratum does not apply to any A15 that is using the standard ACE/AXI interconnect. It only applies to an A15 connected to the CCN-504 Chi interconnect through the SBAS bridge.

If a DVM transaction from the CCN-504 interconnect is sent down the L2 pipeline and the L2 Fill Evict Queue (FEQ) is full, the DVM transaction will be cancelled and reissued. If that DVM transaction also happens to have an address field that matches a copyback (WriteBack, WriteClean, or Evict) in the FEQ, the snoop logic might also detect a hazard that forces a dependency between the DVM and the copyback. Instead of immediately reissuing, the DVM transaction will be held off until the copyback completes.

On a general ACE interconnect this is not a problem, because copyback writes must make forward progress. However, in the Chi interconnect it is possible for that copyback to have a dependency on the DVM transaction completing. The copyback might not complete until the A15 completes the DVM request. This can lead to a system deadlock.

Note: This erratum matches bug #5458 in the ARM internal Jira database.

Configurations affected

Only affects A15 when used with CCN-504

Conditions

1) A DVM request (DVM1) is received on the AR channel from the CCN-504 interconnect.
2) DVM1 is processed when the FEQ is full.
3) DVM1 address matches the address of a WriteBack, WriteClean, or Evict (WB1) in the FEQ WB1 has started to issue, but has not completed sending out its data on the W channel.
4) WB1 is unable to issue its data due to back pressure from the bridge on the write channel. The back pressure will not release until DVM1 completes.

Implications

This erratum might very rarely lead to a system deadlock.

Workaround

To avoid this erratum, A15 configurations used with CCN-504 interconnect must set L2ACTLR[7]. This will enable a timeout mechanism which will allow the DVM request to be reissued even if the copyback has not completed.

When an A15 is used with the CCN-504 interconnect the L2ACTLR must be already configured as specified in the CCN-504 Technical Reference Manual. This workaround sets one additional bit in that register.

2.7. Category C

773023: Order may not be maintained between Strongly Ordered memory requests and Device memory requests on ACE/AXI

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r4p0

Description

A15 maintains order between Strongly Ordered (SO) memory requests as required by the ARM architecture memory ordering model. This is done inside the A15 by use of internal ordering logic. On the interconnect, A15 enforces ordering by using the same ARID/AWID value for all SO memory requests from a given processor (guaranteeing read/read and write/write ordering) and by waiting for the completion of all SO and Device memory requests on one channel before issuing SO or Device requests from the same core on the other channel (guaranteeing read/write and write/read ordering).

A15 does the same for Device memory.

However, A15 uses different ARID and AWID for Device memory requests from a given CPU and Strongly Ordered memory requests from the same CPU. Due to this fact, it is possible that the an SO read from a given CPU could pass a Device read from the same CPU and arrive at a single peripheral out of order.

Conditions

1) A system has memory mapped peripherals larger than 4KB
2) Some of the pages mapped to that peripheral are mapped Strongly Ordered and some are mapped Device
3) Software depends upon ordering of these Strongly Ordered and Device memory requests

Implications

This is not an issue for any device that fits in one 4KB memory page, as it is only possible to have a single memory type for that page (SO/Dev aliasing is not allowed).

For larger peripherals, it is possible that Strongly Ordered or Device transactions could arrive at the peripheral out of order.

Workaround

A given peripheral device should be mapped to all Strongly Ordered or all Device memory.

777769: ICache parity error may not be corrected for NC code

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r4p0

Description

If an instruction fetch to a non-cacheable page in memory gets a false hit on a line in the instruction cache due to a parity error in the instruction cache tag array, incorrect instructions may be executed.

Note: This erratum matches bug #5312 in the ARM internal Jira database.

Conditions

1) MMU enabled
2) Instruction fetch to page marked SO/Dev/Normal-Non-Cacheable
3) Parity error in instruction cache tag
4) Corrupted tag matches the physical address of the cache line being fetched

Implications

In an A15 configured with L1 parity/ECC and with parity/ECC checking enabled, in very rare circumstances a parity error can cause delivery of bad instructions while executing non-cacheable code. This is not expected to be an issue in normal systems as no normal programs will have instructions in non-cacheable memory with the MMU enabled. At boot, or any other time that the MMU is disabled, the erratum will not occur.

Workaround

Place instructions in cacheable memory whenever possible. If you must run non-cacheable code with the MMU enabled, first invalidate the instruction cache.

810072: When a single-bit ECC error occurs in the L2, uncorrected data might be returned

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r4p0.

Description

In the case of an unusual timing boundary condition, a single-bit ECC error in the L2 data array might not be corrected properly.

This erratum cannot occur if ECC is not configured or is not enabled.

Note: This erratum matches bug #5464 in the ARM internal Jira database.

Conditions

1) A cache line fill to address A (CLF_A) hits the L2 and returns data to the L1 data cache.
2) The first 16-byte beat of data returning for CLF_A has a single-bit ECC error in the data. Write streaming is enabled (ACTLR[28:25] != b1111) in the L1.
3) A full cache line-streaming write to address B (WR_B) is ready to issue to the L2.
4) WR_B begins to issue during the same cycle the CLF_A incorrect data returns.
5) A store is in flight (ST_C) to the cache line of address A, or to the cache line being replaced by CLF_A.

Implications

Data corruption might occur if the erratum conditions exist. A single-bit error in the L2 data array might lead to incorrect data in the L1 data cache. A very small percentage of L2 data array single-bit ECC errors will be affected, because the error must be in the critical 16 bytes, must hit a narrow timing window, and must occur when store streaming is pushing full cache line writes to the L2 at the same time as a non-streaming store is hitting the cache line fill or the same set/way.

The result is a very small increase in silent data corruption (SDC) failure in time (FIT) rate in the presence of L2 data array errors.

Workaround

There is no workaround.